Apache Iceberg

Open source table format

Chanukya Pekala DataTribe Collective 23 April 2025

Introduction

Welcome

About me

- Chanukya Pekala
- Data Platform Architect @ Wartsila
- Technical Lead @ DataTribe Collective

Why this topic?

- Table formats are evolving rapidly
- Iceberg is gaining more attention and adoption
- Data Lakes shifts towards open table formats

Agenda

1. Introduction to Apache Iceberg Table Format
2. Iceberg Layout
3. Iceberg Features and Tooling
4. Comparison between Table Formats
5. Iceberg based Data Pipelines
    a. Batch processing
    b. Stream processing
    c. Machine Learning processing

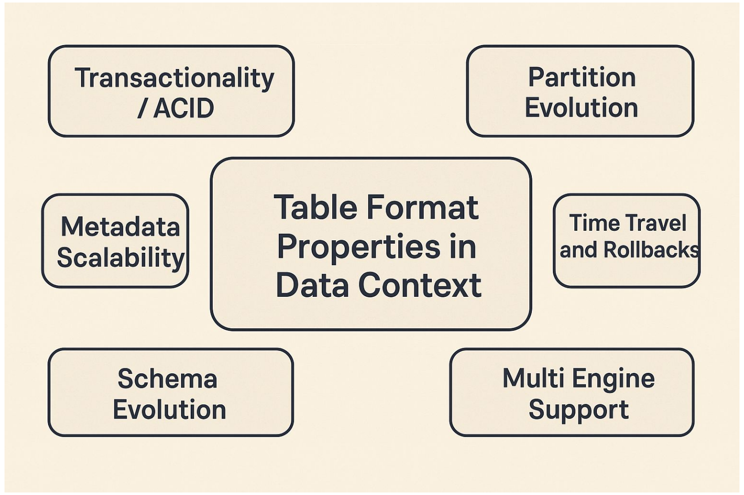
Supermarket example

- Supermarket = data lake
- Signs & maps = metadata
- Aisle = data partition/location
- Oil = data files

When metadata is missing, outdated, or inconsistent, accessing data becomes slow, unreliable, and frustrating — just like shopping in a store with no signage.

Apache Iceberg

- Developed by Netflix, now an Apache project.
- Designed for huge analytic tables (think petabytes).



Its a metadata management layer

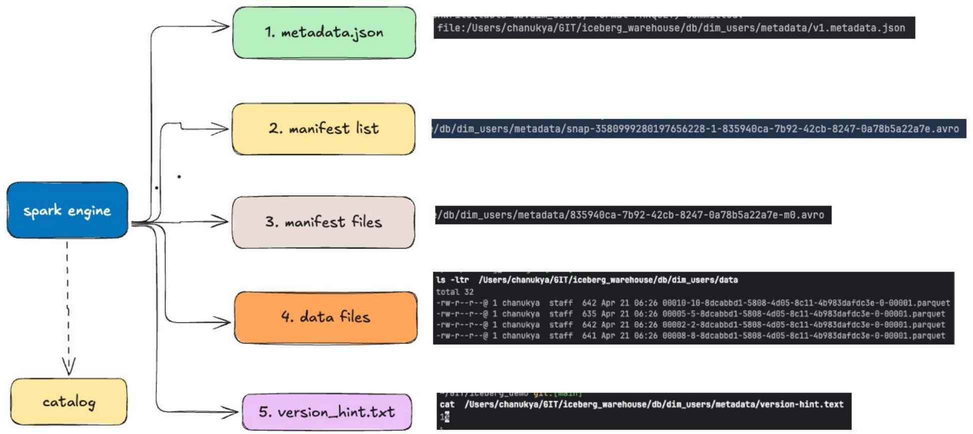
Not a storage layer

Not a compute layer

Not a database

Let's create an iceberg table - local.db.dim_users and then checkout the table layout

Iceberg Layout



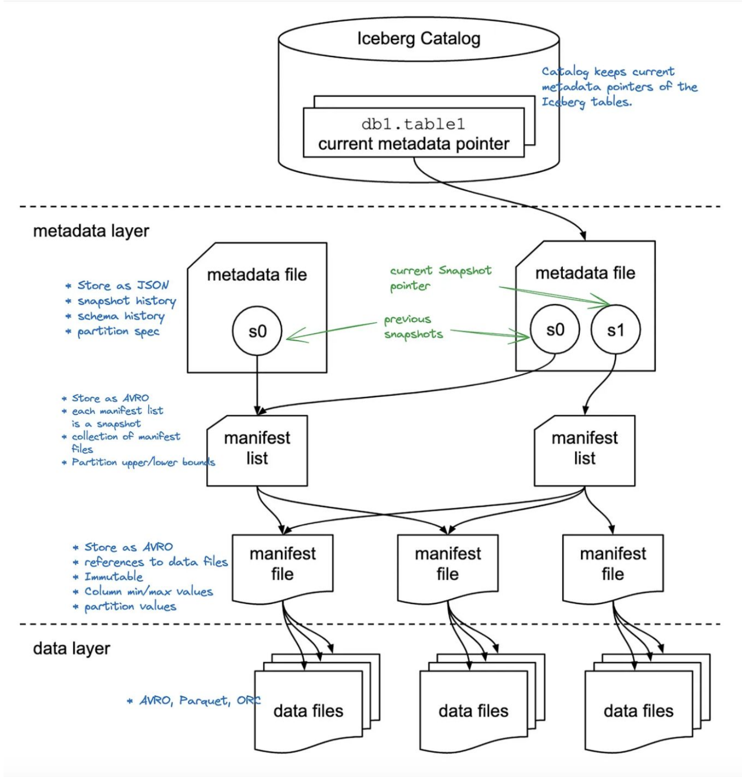
```
file:/Users/chanukya/GIT/iceberg_warehouse/db/dim_users/metadata/v1.metadata.json
```

```
/db/dim_users/metadata/snap-3580999280197656228-1-835940ca-7b92-42cb-8247-0a78b5a22a7e.avro
```

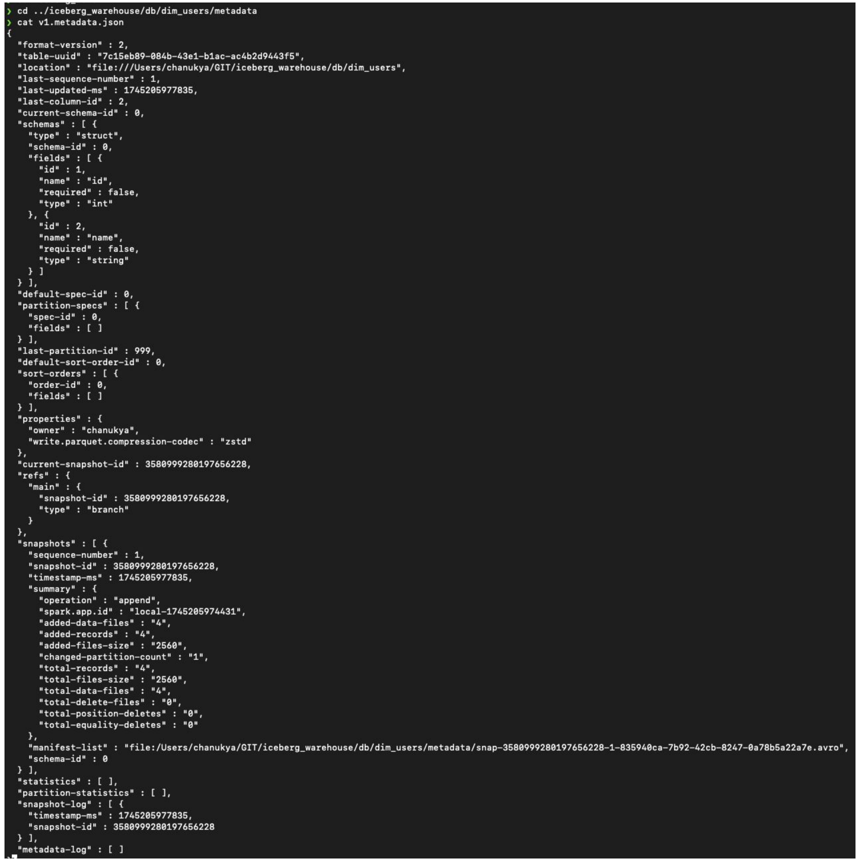
```
/db/dim_users/metadata/835940ca-7b92-42cb-8247-0a78b5a22a7e-m0.avro
```

```
ls -ltr  /Users/chanukya/GIT/iceberg_warehouse/db/dim_users/data
total 32
-rw-r--r--@ 1 chanukya  staff  642 Apr 21 06:26 00010-10-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet
-rw-r--r--@ 1 chanukya  staff  635 Apr 21 06:26 00005-5-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet
-rw-r--r--@ 1 chanukya  staff  642 Apr 21 06:26 00002-2-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet
-rw-r--r--@ 1 chanukya  staff  641 Apr 21 06:26 00008-8-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet
```

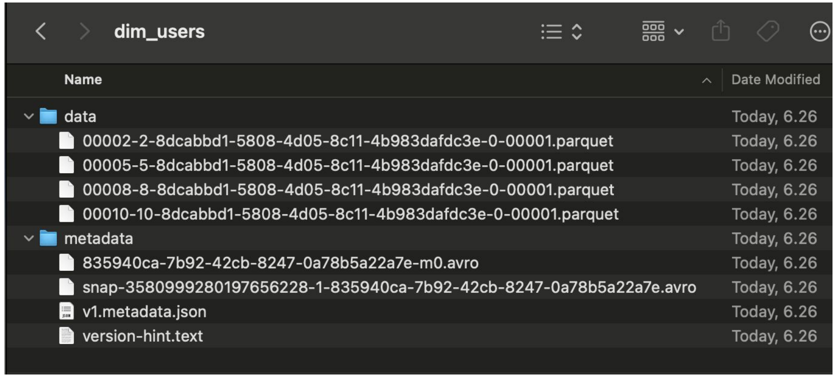
```
cat  /Users/chanukya/GIT/iceberg_warehouse/db/dim_users/metadata/version-hint.text
1
```



metadata.json



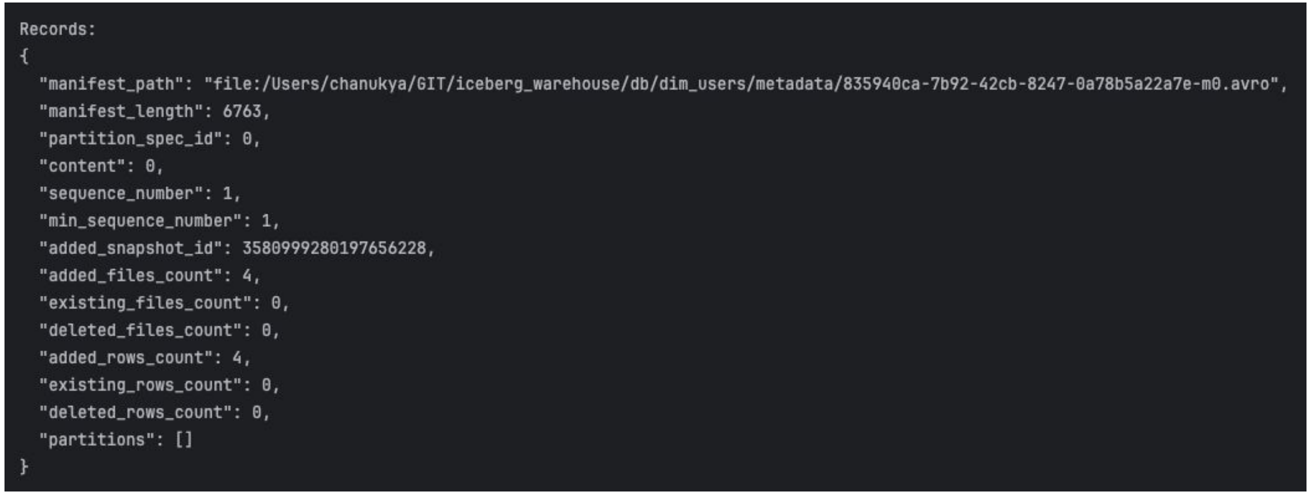
```
❯ cd ../iceberg_warehouse/db/dim_users/metadata
❯ cat v1.metadata.json
{
  "format-version" : 2,
  "table-uuid" : "7c15eb89-084b-43e1-b1ac-ac4b2d9443f5",
  "location" : "file:///Users/chanukya/GIT/iceberg_warehouse/db/dim_users",
  "last-sequence-number" : 1,
  "last-updated-ms" : 1745205977835,
  "last-column-id" : 2,
  "current-schema-id" : 0,
  "schemas" : [ {
    "type" : "struct",
    "schema-id" : 0,
    "fields" : [ {
      "id" : 1,
      "name" : "id",
      "required" : false,
      "type" : "int"
    }, {
      "id" : 2,
      "name" : "name",
      "required" : false,
      "type" : "string"
    } ]
  } ],
  "default-spec-id" : 0,
  "partition-specs" : [ {
    "spec-id" : 0,
    "fields" : [ ]
  } ],
  "last-partition-id" : 999,
  "default-sort-order-id" : 0,
  "sort-orders" : [ {
    "order-id" : 0,
    "fields" : [ ]
  } ],
  "properties" : {
    "owner" : "chanukya",
    "write.parquet.compression-codec" : "zstd"
  },
  "current-snapshot-id" : 3580999280197656228,
  "refs" : {
    "main" : {
      "snapshot-id" : 3580999280197656228,
      "type" : "branch"
    }
  },
  "snapshots" : [ {
    "sequence-number" : 1,
    "snapshot-id" : 3580999280197656228,
    "timestamp-ms" : 1745205977835,
    "summary" : {
      "operation" : "append",
      "spark.app.id" : "local-1745205974431",
      "added-data-files" : "4",
      "added-records" : "4",
      "added-files-size" : "2560",
      "changed-partition-count" : "1",
      "total-records" : "4",
      "total-files-size" : "2560",
      "total-data-files" : "4",
      "total-delete-files" : "0",
      "total-position-deletes" : "0",
      "total-equality-deletes" : "0"
    },
    "manifest-list" : "file:/Users/chanukya/GIT/iceberg_warehouse/db/dim_users/metadata/snap-3580999280197656228-1-835940ca-7b92-42cb-8247-0a78b5a22a7e.avro",
    "schema-id" : 0
  } ],
  "statistics" : [ ],
  "partition-statistics" : [ ],
  "snapshot-log" : [ {
    "timestamp-ms" : 1745205977835,
    "snapshot-id" : 3580999280197656228
  } ],
  "metadata-log" : [ ]
```



```
dim_users

Name                                                                           Date Modified
⌄ data                                                                         Today, 6.26
    00002-2-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet               Today, 6.26
    00005-5-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet               Today, 6.26
    00008-8-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet               Today, 6.26
    00010-10-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet              Today, 6.26
⌄ metadata                                                                     Today, 6.26
    835940ca-7b92-42cb-8247-0a78b5a22a7e-m0.avro                               Today, 6.26
    snap-3580999280197656228-1-835940ca-7b92-42cb-8247-0a78b5a22a7e.avro       Today, 6.26
    v1.metadata.json                                                           Today, 6.26
    version-hint.text                                                          Today, 6.26
```

- Snapshot details
- Partition details
- Schema details
- # of files
- References to manifest-list
- **Benefit** - Centralized table metadata

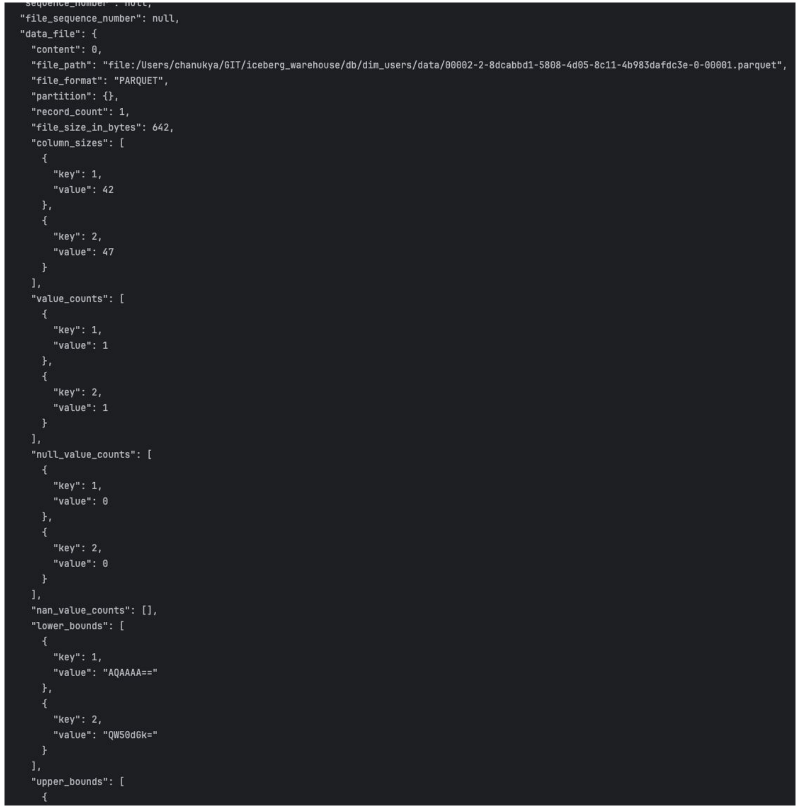
manifest-list



```
Records:
{
  "manifest_path": "file:/Users/chanukya/GIT/iceberg_warehouse/db/dim_users/metadata/835940ca-7b92-42cb-8247-0a78b5a22a7e-m0.avro",
  "manifest_length": 6763,
  "partition_spec_id": 0,
  "content": 0,
  "sequence_number": 1,
  "min_sequence_number": 1,
  "added_snapshot_id": 3580999280197656228,
  "added_files_count": 4,
  "existing_files_count": 0,
  "deleted_files_count": 0,
  "added_rows_count": 4,
  "existing_rows_count": 0,
  "deleted_rows_count": 0,
  "partitions": []
}
```

- Provides manifest file details
- Files added or carried over
- Partitioning range
- File count
- Row counts for the snapshot
- **Benefit**: Lists manifest files associated with a snapshot, enabling efficient tracking of file changes over time.

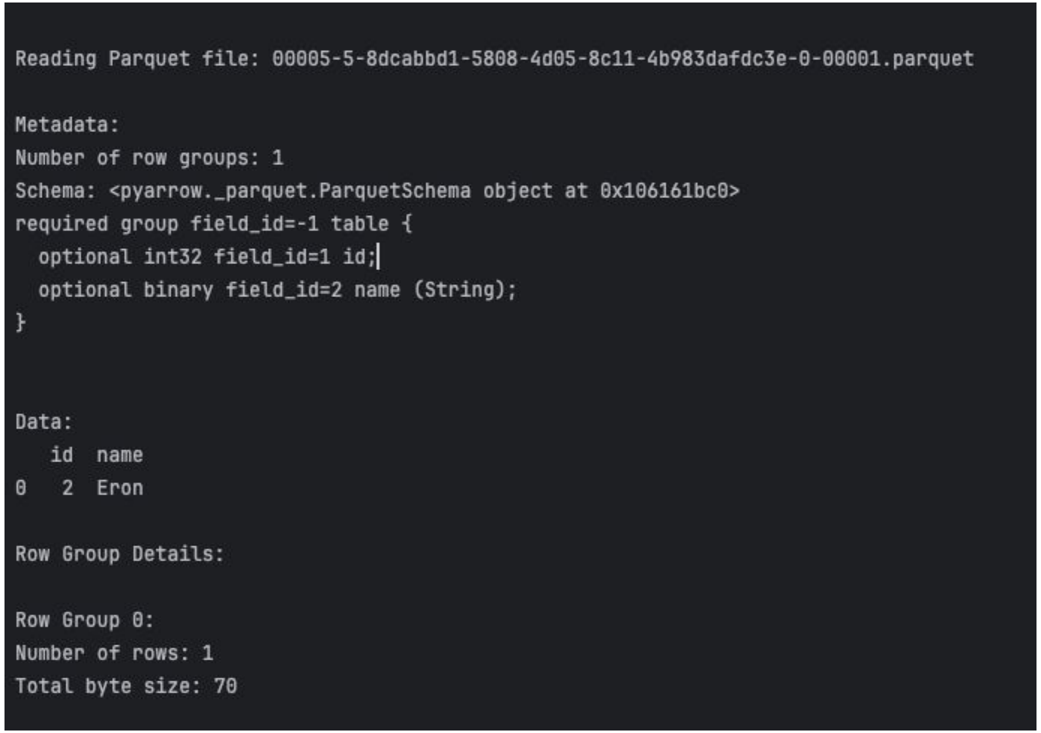
manifest-file



```
  "sequence_number": null,
  "file_sequence_number": null,
  "data_file": {
    "content": 0,
    "file_path": "file:/Users/chanukya/GIT/iceberg_warehouse/db/dim_users/data/00002-2-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet",
    "file_format": "PARQUET",
    "partition": {},
    "record_count": 1,
    "file_size_in_bytes": 642,
    "column_sizes": [
      {
        "key": 1,
        "value": 42
      },
      {
        "key": 2,
        "value": 47
      }
    ],
    "value_counts": [
      {
        "key": 1,
        "value": 1
      },
      {
        "key": 2,
        "value": 1
      }
    ],
    "null_value_counts": [
      {
        "key": 1,
        "value": 0
      },
      {
        "key": 2,
        "value": 0
      }
    ],
    "nan_value_counts": [],
    "lower_bounds": [
      {
        "key": 1,
        "value": "AQAAAA=="
      },
      {
        "key": 2,
        "value": "QW50dGk="
      }
    ],
    "upper_bounds": [
      {
```

- Row counts are file-specific
- Tracks the upper/lower bounds
- Contains references to parquet files.
- **Benefit**: Contains file-level metadata for a set of data files, including column stats, row counts, and partition bounds, for query optimization and file management

data-file

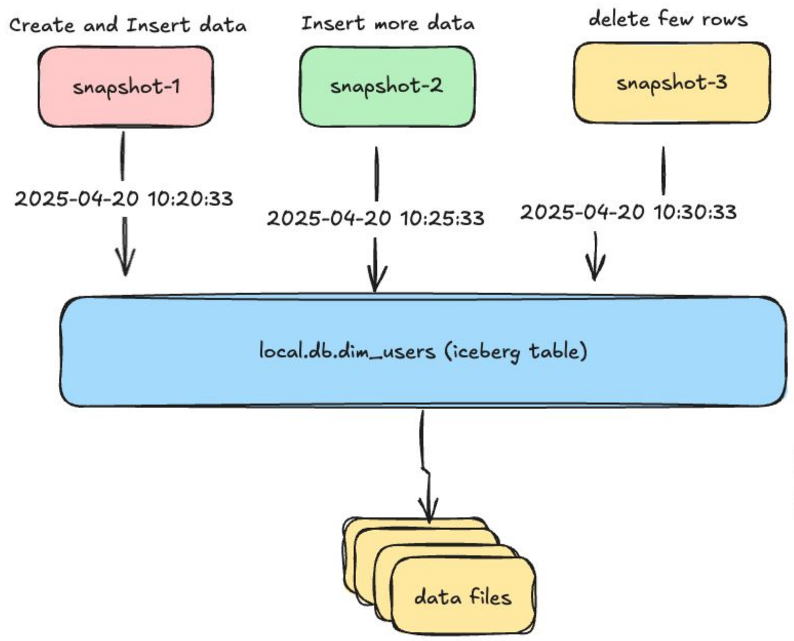


```
Reading Parquet file: 00005-5-8dcabbd1-5808-4d05-8c11-4b983dafdc3e-0-00001.parquet

Metadata:
Number of row groups: 1
Schema: <pyarrow._parquet.ParquetSchema object at 0x106161bc0>
required group field_id=-1 table {
  optional int32 field_id=1 id;
  optional binary field_id=2 name (String);
}


Data:
   id  name
0   2  Eron

Row Group Details:

Row Group 0:
Number of rows: 1
Total byte size: 70
```

- The actual content of the data sits in the underneath parquet files
- **Benefit**: Stores the actual data in a columnar format, with embedded metadata used for efficient querying and predicate pushdown

Few Iceberg Features

Time Travel

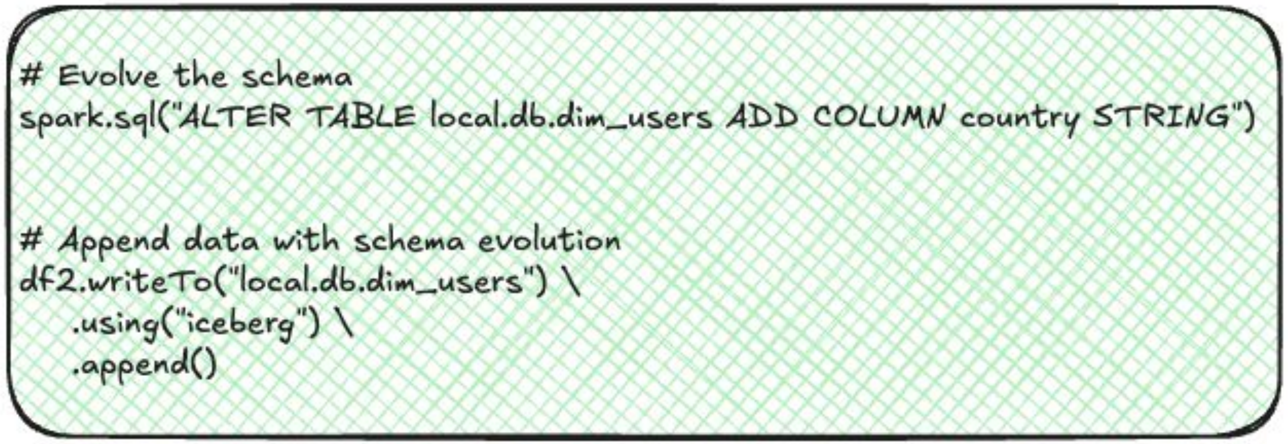


```sql
select * from local.db.dim_users
where version
as of snapshot-1;
```

```sql
select * from local.db.dim_users
where version
as of '2025-04-20 10:20:33';
```

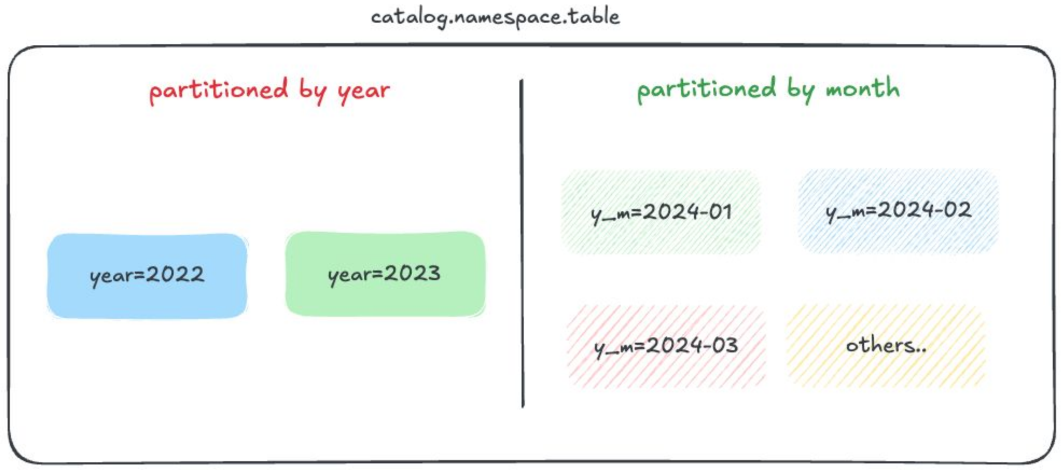
```
+-----------------------+-------------------+-------------------+------------------+
|made_current_at        |snapshot_id        |parent_id |is_current_ancestor|
+-----------------------+-------------------+-------------------+------------------+
|2025-04-20 10:20:33.359|1170971048756548043|NULL  |true  |
|2025-04-20 10:25:33.015|6092563027127301687|1170971048756548043|true  |
+-----------------------+-------------------+-------------------+------------------+
```

Schema Evolution



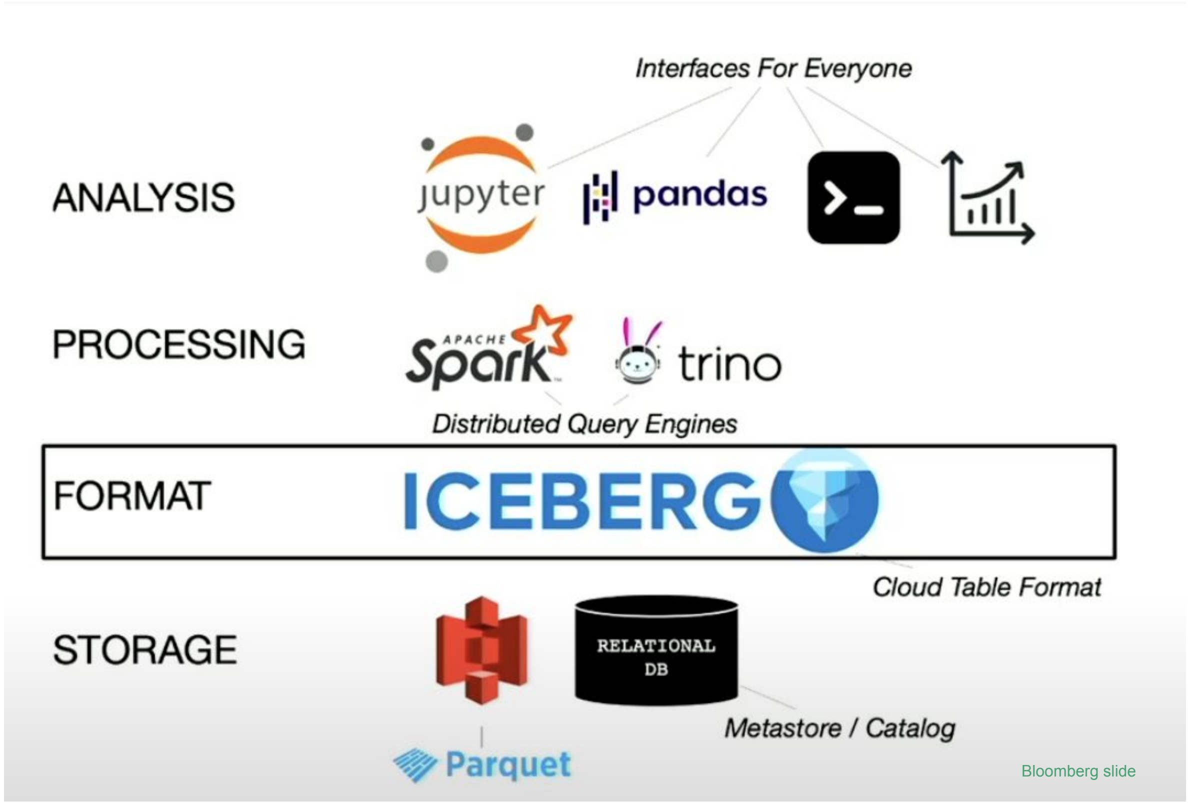
```
# Evolve the schema
spark.sql("ALTER TABLE local.db.dim_users ADD COLUMN country STRING")


# Append data with schema evolution
df2.writeTo("local.db.dim_users") \
    .using("iceberg") \
    .append()
```

Partition Evolution



Currently possible via Java/Scala APIs to replace an existing partition and not available with pySpark 3.5 (subject to correction)

Iceberg Tooling



Popular Lake House Table Formats

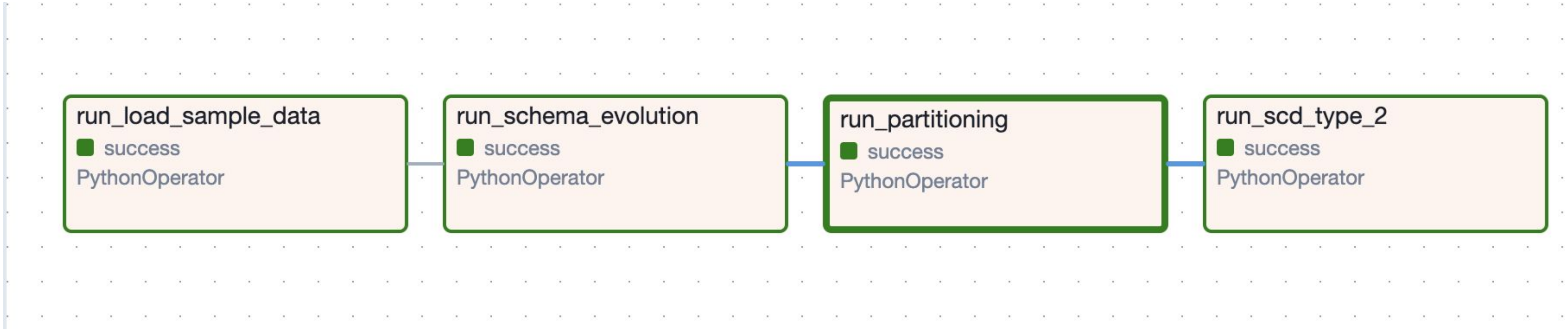
	DELTA LAKE	ICEBERG	Apache hudi
ACID Transactions	Yes - Delta Log + JSON	Yes - Snapshot	Yes
Schema Evolution	Yes	Yes	Yes
Time Travel	Yes - History of changes	Yes - Data versioning	Yes
Partition Evolution	Limited	Yes - Dynamic partition	Yes - flexible
Incremental processing	Batch and Incremental	Batch and Incremental	Batch and Incremental
Primary Focus	Large scale analytics	Large scale analytics Partition management is critical	Frequent updates to tables Real time data ingestion
Base File Format	Parquet	Parquet, ORC and Avro	Parquet and ORC
Major Platform Providers	Databricks	Snowflake, Athena	OneHouse, Uber
Community adoption	High - Databricks ecosystem	Very High - Multi-vendor support	High - Growing adoption
Small File Handling	Auto compaction	Table maintenance API	Auto compaction with cleaning

Other important features

1. Hidden partitioning
2. Table branching and tagging
3. Concurrent write operations
4. External Catalog - Glue, Polaris, Snowflake
5. Table Format Interoperability - Delta to Iceberg, viceversa, XTable

Iceberg based data pipelines

Batch data pipeline



run_load_sample_data
🟩 success
PythonOperator

run_schema_evolution
🟩 success
PythonOperator

run_partitioning
🟩 success
PythonOperator

run_scd_type_2
🟩 success
PythonOperator

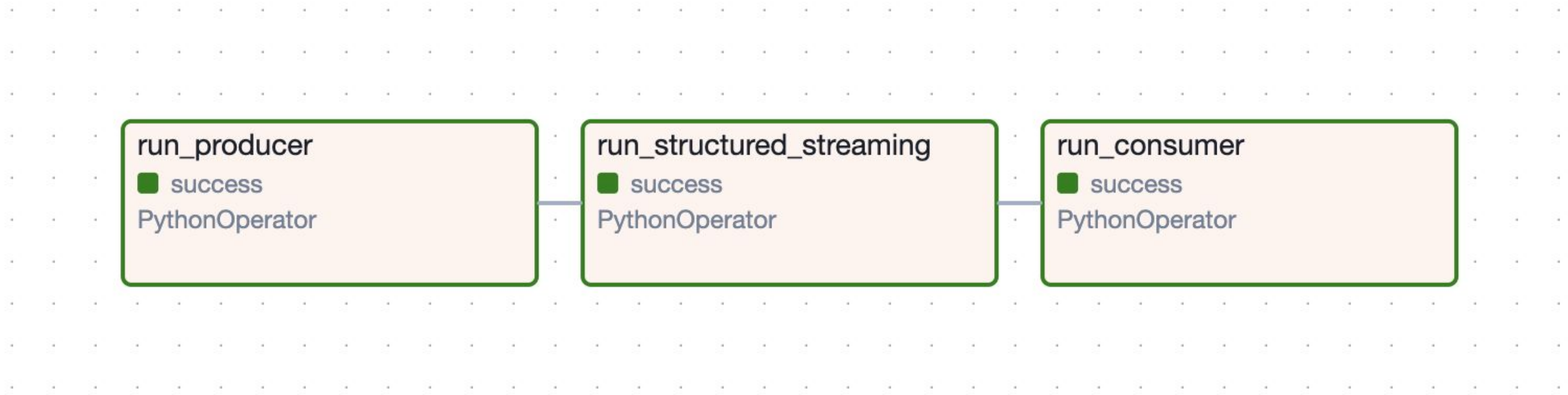
Create local.db.users_dim table (id, name)

Adds a new column (age)

Adds a new column (year), partition table and update partition spec

Add is_current flag and applied merge logic to dim_users_scd2

Stream data pipeline



run_producer
🟩 success
PythonOperator

run_structured_streaming
🟩 success
PythonOperator

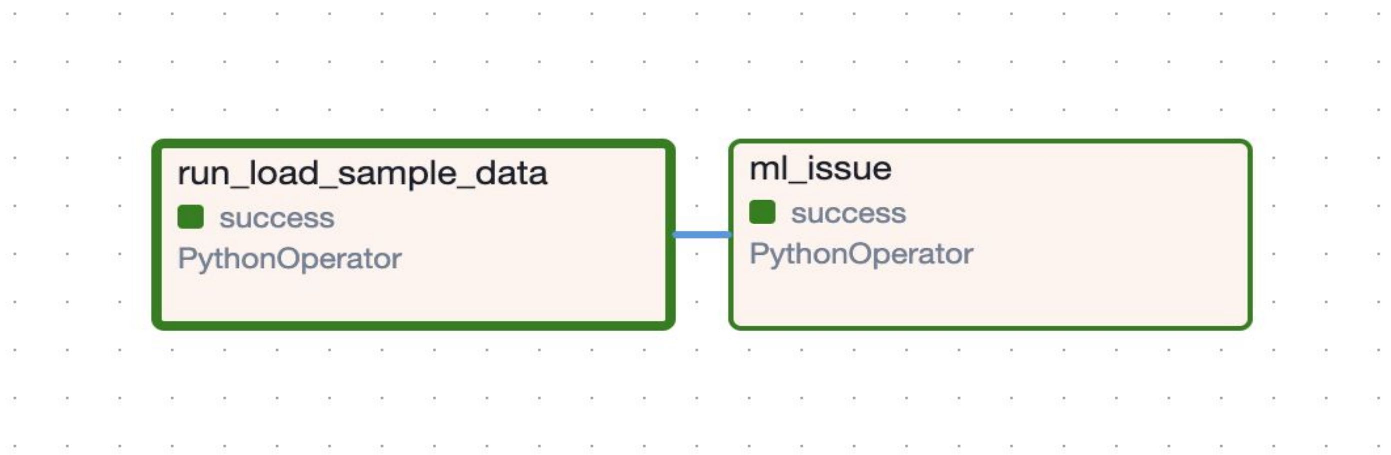
run_consumer
🟩 success
PythonOperator

Writes streaming_records.parquet file

Stream read that file and writes to local.db.structured_streaming_table

Reads streaming_table and writes aggregates to local.db.user_demographics

ML pipeline



Create local.db.users_dim table (id, name)

Creates a linear regression and writes to a new table - local.db.user_segments

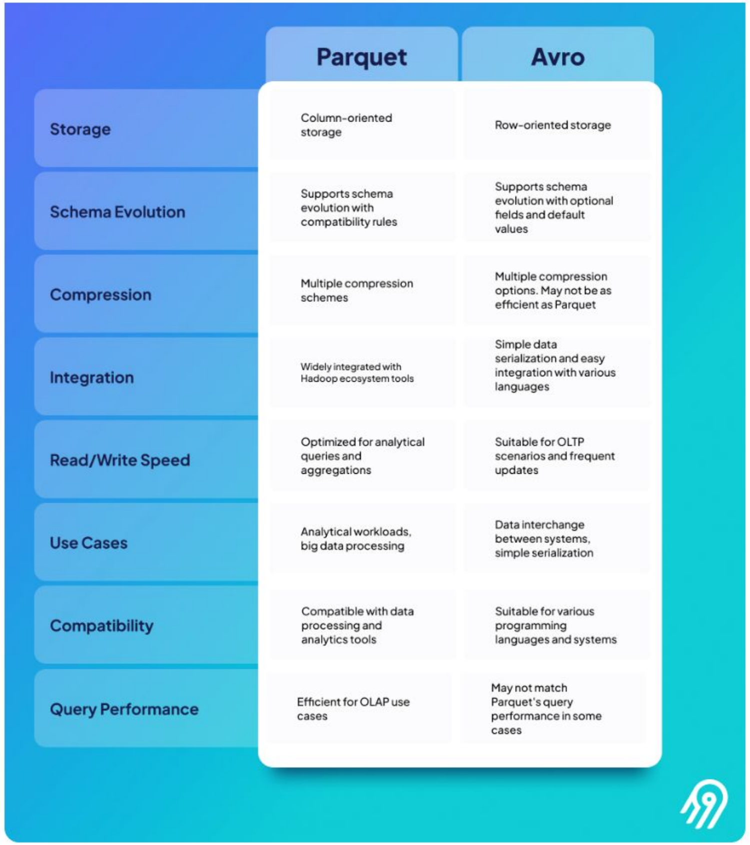
Key takeaways

- Iceberg table metadata layout
- Key features like schema evolution, partitioning, time travel
- Multiple Engines writing towards Iceberg
- Capabilities of the table format - warehousing, machine learning, streaming

Thank you

Q&A...

Supporting Slides



	Parquet	Avro
Storage	Column-oriented storage	Row-oriented storage
Schema Evolution	Supports schema evolution with compatibility rules	Supports schema evolution with optional fields and default values
Compression	Multiple compression schemes	Multiple compression options. May not be as efficient as Parquet
Integration	Widely integrated with Hadoop ecosystem tools	Simple data serialization and easy integration with various languages
Read/Write Speed	Optimized for analytical queries and aggregations	Suitable for OLTP scenarios and frequent updates
Use Cases	Analytical workloads, big data processing	Data interchange between systems, simple serialization
Compatibility	Compatible with data processing and analytics tools	Suitable for various programming languages and systems
Query Performance	Efficient for OLAP use cases	May not match Parquet's query performance in some cases

The main differences between Parquet and Avro

GIT Repo

https://github.com/chanukyapekala/iceberg_demo

Catalog

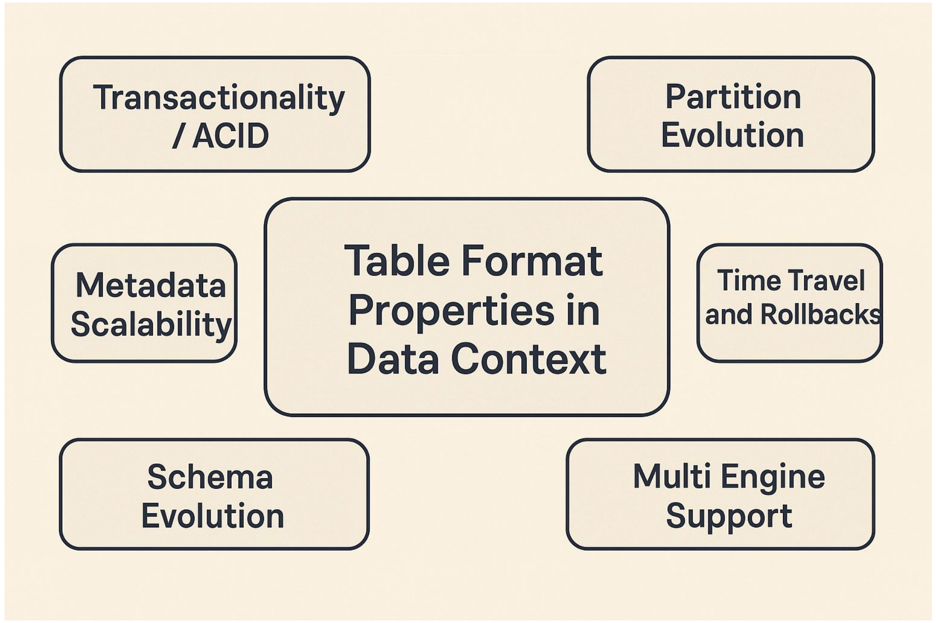
- No catalog
- Lack of version control
- Rules Keep Changing

Problems without table format

Imagine a data lake with just raw files

1. Data discovery becomes hard
2. Data consistency issues
3. No transaction support
4. Slow queries
5. Schema changes are ad hoc and brittle.

Data lakes used to be "write once, read many" systems. But modern analytics needs mutability, versioning, and governance.


Transactionality / ACID
Partition Evolution
Metadata Scalability
Table Format Properties in Data Context
Time Travel and Rollbacks
Schema Evolution
Multi Engine Support

Referred Links

https://medium.com/snowflake/getting-started-with-apache-iceberg-80f338921a31

https://blog.min.io/a-developers-introduction-to-apache-iceberg-using-minio/

https://medium.com/@MarinAgli1/learning-apache-iceberg-an-introspection-f479ee8c7461

https://medium.com/data-engineering-with-dremio/ultimate-directory-of-apache-iceberg-resources-e3e02efac62e